ELSEVIER

Theoretical Computer Science 297 (2003) 281–295

Theoretical Computer Science

www.elsevier.com/locate/tcs

Almost k-wise independence and hard Boolean functions

Valentine Kabanets

Department of Computer Science, University of Toronto, Toronto, Ont., Canada

Abstract

We construct Boolean functions (computable by polynomial-size circuits) with large lower bounds for read-once branching program (1-b.p.'s): a function in P with the lower bound $2^{n-\mathrm{polylog}(n)}$, a function in quasipolynomial time with the lower bound $2^{n-\mathrm{O}(\log n)}$, and a function in LINSPACE with the lower bound $2^{n-\log n-\mathrm{O}(1)}$. Our constructions are simpler than those of Andreev et al. (Electronic Colloq. on Computational Complexity, TR97-053, 1997), as we apply the idea of almost k-wise independence more directly, without the use of discrepancy set generators for large affine subspaces. The simplicity of our constructions also allows us to observe that there exists a Boolean function in $\mathrm{AC}^0[2]$ (computable by a depth 3, polynomial-size circuit over the basis $\{\wedge,\oplus,1\}$) with the optimal lower bound $2^{n-\log n-\mathrm{O}(1)}$ for 1-b.p.'s.

Keywords: Almost k-wise independence; Derandomization; Exponential lower bounds for read-once branching programs; r-Mixed Boolean functions

1. Introduction

The main goal of the computational complexity theory is to show that certain natural Boolean functions are hard. For example, the negative solution to the famous "$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$" question is equivalent to showing that SAT, the problem of deciding whether a given propositional logical formula is satisfiable, is hard for the class P. In general, the notion of hardness of a Boolean function is dependent on a chosen model of computation. One can talk about Boolean functions that are hard for resource-bounded Turing machines (the uniform model), or for size-bounded Boolean circuits (the nonuniform model). Determining the relationship between uniform and nonuniform hardness is an important

E-mail address: kabanets@cs.toronto.edu (V. Kabanets).

0304-3975/03/$ - see front matter
PII: S0304-3975(02)00643-6

open problem whose solution would have significant consequences for theoretical computer science.

To be more concrete, let us consider the uniform complexity class $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ of all languages decidable in linear-exponential time by a deterministic Turing machine, and the nonuniform complexity class $\mathrm{SIZE}(2^n/n)$ of all n-variable Boolean functions decidable by a Boolean circuit of size at most $2^n/n$. It is well known that most n-variable Boolean functions are outside $\mathrm{SIZE}(2^n/n)$ [18]. Is there a family of Boolean functions $f = \{f_n\}_{n \geqslant 0}$ such that $f \notin \mathrm{SIZE}(2^n/n)$ but $f \in \mathrm{E}$? If the answer is negative, then $\mathrm{P} \neq \mathrm{NP}$ (by a rather simple argument). If the answer is positive, then $\mathrm{BPP} = \mathrm{P}$ by a beautiful theorem of Impagliazzo and Wigderson [7]. At present, however, we cannot even decide if every Boolean function in E is computable by a family of linear-size circuits.

A natural approach towards proving that a given Boolean function is nonuniformly hard is to exhibit a certain combinatorial property of Boolean functions that is true only for hard Boolean functions, and then show that the given Boolean function satisfies this property. As Razborov and Rudich [14] argue, this approach may be flawed since such combinatorial properties may not exist for the general nonuniform model of computation. However, such hardness properties do exist for the case of restricted models, such as read-once branching programs. Below, we shall use the existence of this property to show that, e.g., the uniform class LINSPACE of all languages decidable by a linear-space bounded deterministic Turing machine contains a language of maximal, to within a constant factor, size for read-once branching programs. First, we will observe that a random Boolean function satisfies the 1-b.p. hardness property with high probability. Then we will construct an efficient function generator that "fools" this hardness property in the sense that a random Boolean function output by the generator also satisfies this property with high probability. Finally, the desired hard Boolean function will be the lexicographically first function f output by the generator such that f satisfies the 1-b.p. hardness property.

Branching programs were introduced in order to measure the space complexity of Turing machines with advice. Recall that a *branching program* is a directed acyclic graph with one source and with each node of out-degree at most 2. Each node of out-degree 2 (a branching node) is labeled by an index of an input bit, with one outgoing edge labeled by 0, and the other by 1; each node of out-degree 0 (a sink) is labeled by 0 or 1. The branching program accepts an input if there is a path from the source to a sink labeled by 1 such that, at each branching node of the path, the path contains the edge labeled by the input bit for the input index associated with that node. Finally, the *size* of a branching program is defined as the number of its nodes.

While there are no nontrivial lower bounds on the size of general branching programs, strong lower bounds were obtained for a number of explicit Boolean functions in restricted models (see, e.g., [13] for a survey). In particular, for *read-once branching programs* (1-b.p.'s)—where, on every path from the source to a sink, no two branching nodes are labeled by the same input index—exponential lower bounds of the form $2^{\Omega(\sqrt{n})}$ were given for explicit n-variable Boolean functions in [4–6,8,9,11,19–21] among others. Moreover, [8,9,6,4] exhibited Boolean functions in AC^0 that require 1-b.p.'s of size at least $2^{\Omega(\sqrt{n})}$.

After lower bounds of the form $2^{\Omega(\sqrt{n})}$ were obtained for 1-b.p.'s, the natural problem was to find an explicit Boolean function with the truly exponential lower bound $2^{\Omega(n)}$. The first such bound was proved in [1] for the Boolean function computing the parity of the number of triangles in a graph; the constant factor was later improved in [19]. With the objective to improve this lower bound, Savický and Žák [17] constructed a Boolean function in P that requires a 1-b.p. of size at least $2^{n-3\sqrt{n}}$, and gave a probabilistic construction of a Boolean function requiring a 1-b.p. of size at least $2^{n-O(\log n)}$. Finally, Andreev et al. [3] presented a Boolean function in LINSPACE$\cap$P/poly with the optimal lower bound $2^{n-\log n+O(1)}$, and, by derandomizing the probabilistic construction in [17], a Boolean function in QP$\cap$P/poly with the lower bound $2^{n-O(\log n)}$, as well as a Boolean function in P with the lower bound $2^{n-\text{polylog}(n)}$; here QP stands for the quasipolynomial time $n^{\text{polylog}(n)}$.

The combinatorics of 1-b.p.'s is quite well understood: a theorem of Simon and Szegedy [19], generalizing the ideas of many papers on the subject, provides a way of obtaining strong lower bounds. A particular case of this theorem states that any 1-b.p. computing an r-mixed Boolean function has size at least $2^r - 1$. Informally, an r-mixed function essentially depends on every set of r variables (see the next section for a precise definition). The reason why this lower-bound criterion works can be summarized as follows. A subprogram of a 1-b.p. G_n starting at a node v does not depend on any variable queried along any path going from the source s of G_n to v, and hence v completely determines a subfunction of the function computed by G_n. If G_n computes an r-mixed Boolean function f_n, then any two paths going from s to v can be shown to query the same variables, whenever v is sufficiently close to s. Hence, such paths must coincide, i.e., assign the same values to the queried variables; otherwise, two different assignments to a set of at most r variables yield the same subfunction of f_n, contradicting the fact that f_n is r-mixed. It follows that, near the source, G_n is a complete binary tree, and so it must have exponentially many nodes.

Andreev et al. [3] construct a Boolean function $f_n(x_1,\ldots,x_n)$ in LINSPACE$\cap$P/poly that is r-mixed for $r = n - \lceil \log n \rceil - 2$ for almost all n. By the lower-bound criterion mentioned above, this yields the optimal lower bound $\Omega(2^n/n)$ for 1-b.p.'s. A Boolean function in DTIME$(2^{\log^2 n}) \cap$ P/poly that requires a 1-b.p. of size at least $2^{n-O(\log n)}$ is constructed by reducing the amount of randomness used in the probabilistic construction of [17] to $O(\log^2 n)$ advice bits. Since these bits turn out to determine a polynomial-time computable function with the lower bound $2^{n-O(\log n)}$, one gets a function in P with the lower bound $2^{n-O(\log^2 n)}$ by making the advice bits a part of the input.

Both constructions in [3] use the idea of ε-biased sample spaces introduced by Naor and Naor [10], who also gave an algorithm for generating small sample spaces; three simpler constructions of such spaces were later given by Alon et al. [2]. Andreev et al. define certain ε-discrepancy sets for systems of linear equations over GF(2), and relate these discrepancy sets to the biased sample spaces of Naor and Naor through a reduction lemma. Using a particular construction of a biased sample space (the powering construction from [2]), Andreev et al. give an algorithm for generating ε-discrepancy sets, which is then used to derandomize both a probabilistic construction of an r-mixed Boolean function for $r = n - \lceil \log n \rceil - 2$ and the construction in [17] mentioned above.

1.1. Our results

We will show that the known algorithms for generating small ε-biased sample spaces can be applied *directly* to get the r-mixed Boolean function as above, and to derandomize the construction in [17]. The idea of our first construction is very simple: treat the elements (bit strings) of an ε-biased sample space as the truth tables of Boolean functions. This will induce a probability distribution on Boolean functions such that, on any subset A of k inputs, the restriction to A of a Boolean function chosen according to this distribution will look almost as if it were a uniformly chosen random function defined on the set A. By an easy counting argument, we will show that such a space of functions will contain the desired r-mixed function, for a suitable choice of parameters ε and k.

We indicate several ways of obtaining an r-mixed Boolean function with $r = n - \lceil \log n \rceil - 2$. In particular, using Razborov's construction of ε-biased sample spaces that are computable by $AC^0[2]$ formulas [12] (see also [15]), we prove that there are such r-mixed functions that belong to the class of polynomial-size depth 3 formulas over the basis $\{\&, \oplus, 1\}$. This yields the smallest (nonuniform) complexity class known to contain Boolean functions with the optimal lower bounds for 1-b.p.'s. (We remark that, given our lack of strong circuit lower bounds, it is conceivable that the characteristic function of every language in EXP can be computed in nonuniform $AC^0[6]$.)

In our second construction, we derandomize a probabilistic existence proof in [17]. We proceed along the usual path of derandomizing probabilistic algorithms whose analysis depends only on almost k-wise independence rather than full independence of random bits [10]. Observing that the construction in [17] is one such algorithm, we reduce its randomness complexity to $O(\log^3 n)$ bits (again treating strings of an appropriate sample space as truth tables). This gives us a $\mathrm{DTIME}(2^{O(\log^3 n)})$-computable Boolean function of quasilinear circuit-size with the lower bound for 1-b.p.'s slightly better than that for the corresponding quasipolynomial-time computable function in [3], and a Boolean function in quasilinear time, QL, with the lower bound for 1-b.p.'s at least $2^{n-O(\log^3 n)}$, which is only slightly worse than the lower bound for the corresponding polynomial-time function in [3]. In the analysis of our construction, we employ a combinatorial lemma due to Razborov [12], which bounds from above the probability that none of n events occur, given that these events are almost k-wise independent.

1.2. The remainder of the paper

In the following section, we state the necessary definitions and some auxiliary lemmas. In Section 3, we show how to construct an r-mixed function that has the same optimal lower bound for 1-b.p. as that in [3], and observe that such a function can be computed in $AC^0[2]$. In Section 4, we give a simple derandomization procedure for a construction in [17], obtaining two more Boolean functions (computable in polynomial time and quasipolynomial time, respectively) that are hard with respect to 1-b.p.'s.

In Section 5, we use our techniques to show that the class QP contains $n - O(\log n)$-mixed Boolean functions. Finally, we give concluding remarks in Section 6.

2. Preliminaries

Below, we recall the standard definitions of k-wise independence and (ε, k)-independence. We consider probability distributions that are uniform over some set $S \subseteq \{0,1\}^n$; such a set is denoted by S_n and called a *sample space*.

Let S_n be a sample space, and let $X = x_1 \cdots x_n$ be a string chosen uniformly from S_n. Then S_n is *k-wise independent* if, for any k indices $i_1 < i_2 < \cdots < i_k$ and any k-bit string α, we have $\Pr[x_{i_1} x_{i_2} \cdots x_{i_k} = \alpha] = 2^{-k}$. Similarly, for S_n and X as above, S_n is *(ε, k)-independent* if $|\Pr[x_{i_1} x_{i_2} \cdots x_{i_k} = \alpha] - 2^{-k}| \leqslant \varepsilon$ for any k indices $i_1 < i_2 < \cdots < i_k$ and any k-bit string α.

Naor and Naor [10] present an efficient construction of small (ε, k)-independent sample spaces; three simpler constructions are given in [2]. Here we recall just one construction from [2], the powering construction, although any of their three constructions could be used for our purposes.

Consider the Galois field $\mathrm{GF}(2^m)$ and the associated m-dimensional vector space over $\mathrm{GF}(2)$, where the field operations in this space are performed modulo the lexicographically first [1] irreducible polynomial of degree m over $\mathrm{GF}(2)$. For every element u of $\mathrm{GF}(2^m)$, let $\mathrm{bin}(u)$ denote the corresponding binary vector in the associated vector space. The sample space Pow_N^{2m} is defined as a set of N-bit strings such that each string ω is determined as follows. Two elements $x, y \in \mathrm{GF}(2^m)$ are chosen uniformly at random. For each $1 \leqslant i \leqslant N$, the ith bit ω_i is defined as $\langle \mathrm{bin}(x^i), \mathrm{bin}(y) \rangle$, where $\langle a, b \rangle$ denotes the inner product over $\mathrm{GF}(2)$ of binary vectors a and b.

The next lemma follows from the results in [2] (Proposition 3 and Corollary 1).

Lemma 1 (Alon et al. [2]). *The sample space* Pow_N^{2m} *is* $(N/2^m, k)$*-independent for every* $k \leqslant N$.

As we have mentioned in the introduction, we shall view the strings of the sample space Pow_N^{2m} as the truth tables of Boolean functions of $\log N$ variables. It will be convenient to assume that N is a power of 2, i.e., $N = 2^n$. Thus, the uniform distribution over the sample space $\mathrm{Pow}_{2^n}^{2m}$ induces a distribution on Boolean functions of n variables. More formally, we define a function generator that associates with each pair of strings $x, y \in \{0,1\}^m$ an n-variable Boolean function

$$F_{x,y}(z) = \langle \mathrm{bin}(x^{\bar{z}}), \mathrm{bin}(y) \rangle,$$

where $\bar{z}$ is a natural number whose binary representation is $z \in \{0,1\}^n$. Observe that, for every x and y, the Boolean function $F_{x,y}(z)$ is polynomial-time computable, provided

[1] We have chosen the lexicographically first irreducible polynomial just to be concrete; any other irreducible polynomial would also work.

that we are given the coefficients of the lexicographically first irreducible degree-m polynomial over GF(2). Our function generator satisfies the following lemma.

Lemma 2. *Let A be any set of k strings from $\{0,1\}^n$, for any $k \leqslant 2^n$, and let ϕ be any Boolean function defined on A. If $f = F_{x,y}$ for $x, y \in \{0,1\}^m$ chosen uniformly at random, then $|\Pr[f|_A = \phi] - 2^{-k}| \leqslant 2^{-(m-n)}$, where $f|_A$ denotes the restriction of f to the set A.*

Proof. The k strings in A determine k indices $i_1, \ldots, i_k$ in the truth table of f. The function ϕ is determined by its truth table, a binary string α of length k. Now the claim follows immediately from Lemma 1 and the definition of (ε, k)-independence. □

Razborov [12] showed that there exist complex combinatorial structures (such as the Ramsey graphs, rigid graphs, etc.) of exponential size which can be encoded by polynomial-size bounded-depth Boolean formulas over the basis $\{\&, \oplus, 1\}$. In effect, Razborov gave a construction of ε-biased sample spaces (using the terminology of [10]), where the elements of such sample spaces are the truth tables of $\mathrm{AC}^0[2]$-computable Boolean functions chosen according to a certain distribution on $\mathrm{AC}^0[2]$-formulas. We describe this distribution next.

For $n, m, l \in \mathbb{N}$, a random formula $\mathbf{F}(n, m, l)$ of depth 3 is defined as

$$\mathbf{F}(n,m,l) = \bigoplus_{\alpha=1}^{l} \bigwedge_{\beta=1}^{m} \left(\left(\bigoplus_{\gamma=1}^{n} \lambda_{\alpha\beta\gamma} x_\gamma \right) \oplus \lambda_{\alpha\beta} \right), \tag{1}$$

where $\{\lambda_{\alpha\beta}, \lambda_{\alpha\beta\gamma}\}$ is a collection of $(n+1)ml$ independent random variables uniformly distributed on $\{0,1\}$. The following lemma shows that this distribution determines an ε-biased sample space; as observed in [15], a slight modification of the above construction yields somewhat better parameters, but the simpler construction would suffice for us here.

Lemma 3 (Razborov [12]). *Let $k, l, m \in \mathbb{N}$ be any numbers such that $k \leqslant 2^{m-1}$, let A be any set of k strings from $\{0,1\}^n$, and let ϕ be any Boolean function defined on A. For a Boolean function f computed by the random formula $\mathbf{F}(n,m,l)$ defined in (1), we have $|\Pr[f|_A = \phi] - 2^{-k}| \leqslant \mathrm{e}^{-l2^{-m}}$, where $f|_A$ denotes the restriction of f to the set A.*

The proof of Lemma 3 is most easily obtained by manipulating certain discrete Fourier transforms. We refer the interested reader to [12] or [15] for details.

Below, we give the definitions of some classes of Boolean functions hard for 1-b.p.'s. We say that a Boolean function $f_n(x_1, \ldots, x_n)$ is *r-mixed* for some $r \leqslant n$ if, for every subset X of r input variables $\{x_{i_1}, \ldots, x_{i_r}\}$, no two distinct assignments to X yield the same subfunction of f in the remaining $n - r$ variables. We shall see in the following section that an r-mixed function for $r = n - \lceil \log n \rceil - 2$ has a nonzero probability in a distribution $\mathbf{F}_{n,m}$, where $m \in \mathrm{O}(n)$, and in the distribution induced by the random formula $\mathbf{F}(n,m,l)$, where $m \in \mathrm{O}(\log n)$ and $l \in \mathrm{poly}(n)$.

It was observed by many researchers that r-mixed Boolean functions are hard for 1-b.p.'s. The following lemma is implicit in [20,5], and is a particular case of results in [8,19].

Lemma 4 (Wegener [20], Dunne [5], Jukna [8], Simon and Szegedy [19]). *Let $f_n(x_1,\ldots,x_n)$ be an r-mixed Boolean function, for some $r \leqslant n$. Then every 1-b.p. computing f_n has size at least $2^r - 1$.*

Following Savický and Žák [17], we call a function $\phi : \{0,1\}^n \to \{1,2,\ldots,n\}$ (s,n,q)-*complete*, for some integers s, n, and q, if for every set $I \subseteq \{1,\ldots,n\}$ of size $n-s$ we have

1. for every 0–1 assignment to the variables x_i, $i \in I$, the range of the resulting subfunction of ϕ is equal to $\{1,2,\ldots,n\}$, and
2. there are at most q different subfunctions of ϕ, as one varies over all 0–1 assignments to x_i, $i \in I$.

Our interest in (s,n,q)-complete functions is justified by the following lemma; its proof is based on a generalization of Lemma 4.

Lemma 5 (Savicky and Zak [17]). *Let $\phi : \{0,1\}^n \to \{1,2,\ldots,n\}$ be an (s,n,q)-complete function. Then the Boolean function $f_n(x_1,\ldots,x_n) = x_{\phi(x_1,\ldots,x_n)}$ requires 1-b.p.'s of size at least $2^{n-s}/q$.*

The following lemma can be used to construct an (s,n,q)-complete function.

Lemma 6 (Savicky and Zak [17]). *Let A be a $t \times n$ matrix over GF(2) with every $t \times s$ submatrix of rank at least r. Let $\psi : \{0,1\}^t \to \{1,2,\ldots,n\}$ be a mapping such that its restriction to every affine subset of $\{0,1\}^t$ of dimension at least r has the range $\{1,2,\ldots,n\}$. Then the function $\phi(x) = \psi(Ax)$ is $(s,n,2^t)$-complete.*

A probabilistic argument shows that a $t \times n$ matrix A and a function $\psi : \{0,1\}^t \to \{1,2,\ldots,n\}$ exist that satisfy the assumptions of Lemma 6 for the choice of parameters $s,t,r \in \mathrm{O}(\log n)$, thereby yielding a Boolean function that requires 1-b.p.'s of size at least $2^{n-\mathrm{O}(\log n)}$. Below we will show that the argument uses only limited independence of random bits, and hence it can be derandomized using the known constructions of (ε,k)-independent spaces. We will utilize the following lemma due to Razborov; for completeness, we provide its proof.

Lemma 7 (Razborov [12]). *Let $l > 2k$ be any natural numbers, let $0 < \theta$, $\varepsilon < 1$, and let $\mathscr{E}_1,\ldots,\mathscr{E}_l$ be events such that, for every subset $I \subseteq \{1,\ldots,l\}$ of size at most k, we have $|\Pr[\bigwedge_{i\in I} \mathscr{E}_i] - \theta^{|I|}| \leqslant \varepsilon$. Then $\Pr[\bigwedge_{i=1}^{l} \mathscr{E}_i] \leqslant \mathrm{e}^{-\theta l} + \binom{l}{k+1}(\varepsilon k + \theta^k)$.*

Proof. We first consider the case where k is even. Let $\mathscr{C}_1,\ldots,\mathscr{C}_l$ be independent events, each having the success probability θ. Applying the Boole–Bonferroni inequality to

$\Pr[\bigvee_{i=1}^{l} \mathscr{E}_i]$ and $\Pr[\bigvee_{i=1}^{l} \mathscr{C}_i]$, we obtain that

$$\Pr\left[\bigvee_{i=1}^{l} \mathscr{E}_i\right] \geqslant \sum_{v=1}^{k} (-1)^{v+1} \sum_{|I|=v} \Pr\left[\bigwedge_{i\in I} \mathscr{E}_i\right] \tag{2}$$

and

$$\Pr\left[\bigvee_{i=1}^{l} \mathscr{C}_i\right] \leqslant \sum_{v=1}^{k} (-1)^{v+1} \sum_{|I|=v} \theta^{|I|} + \sum_{|I|=k+1} \theta^{k+1}. \tag{3}$$

The assumption of the lemma that $\mathscr{E}_1,\ldots,\mathscr{E}_l$ are almost k-wise independent implies that the right-hand side in (2) is at least

$$\sum_{v=1}^{k} (-1)^{v+1} \sum_{|I|=v} \theta^{|I|} - \varepsilon k \binom{l}{k}. \tag{4}$$

On the other hand, the independence of $\mathscr{C}_1,\ldots,\mathscr{C}_l$ implies that

$$\Pr\left[\bigvee_{i=1}^{l} \mathscr{C}_i\right] = 1-(1-\theta)^l \geqslant 1-\mathrm{e}^{-\theta l}. \tag{5}$$

Combining (2)–(5) yields (for even k) that

$$\begin{aligned}\Pr\left[\bigvee_{i=1}^{l} \mathscr{E}_i\right] &\geqslant 1-\mathrm{e}^{-\theta l} - \varepsilon k \binom{l}{k} - \theta^{k+1}\binom{l}{k+1} \\ &\geqslant 1-\mathrm{e}^{-\theta l} - \binom{l}{k+1}(\varepsilon k + \theta^{k+1}).\end{aligned}$$

In the case where k is odd, we use the above argument with $k-1$ substituted for k. This completes the proof. □

3. Constructing $(n - \lceil \log n \rceil - 2)$-mixed Boolean functions

First, we give a simple probabilistic argument showing that r-mixed functions exist for $r = n - \lceil \log n \rceil - 2$. Let f be a Boolean function on n variables that is chosen uniformly at random from the set of all Boolean n-variable functions. For any fixed set of indices $\{i_1,\ldots,i_r\} \subseteq \{1,\ldots,n\}$ and any two fixed binary strings $\alpha = \alpha_1,\ldots,\alpha_r$ and $\beta = \beta_1,\ldots,\beta_r$, the probability that fixing $x_{i_1},\ldots,x_{i_r}$ to α and then to β will give the same subfunction of f in the remaining $n-r$ variables is 2^{-k}, where $k = 2^{n-r}$. Thus, the probability that f is not r-mixed is at most $\binom{n}{r}2^{2r}2^{-k}$, which tends to 0 as n grows.

We observe that the above argument only used the fact that f is random on any set of $2k$ inputs: those obtained after the r variables $x_{i_1},\ldots,x_{i_r}$ are fixed to α, the set of

which will be denoted as A_α, plus those obtained after the same variables are fixed to β, the set of which will be denoted as A_β. This leads us to the following theorem.

Theorem 8. *There is an $m \in O(n)$ such that the probability that a Boolean n-variable function $f = F_{x,y}$, for $x, y \in \{0,1\}^m$ chosen uniformly at random, is r-mixed for $r = n - \lceil \log n \rceil - 2$ tends to 1 as n grows.*

Proof. By Lemma 2, a random Boolean function $f = F_{x,y}$ is equal to any fixed Boolean function ϕ defined on a set $A_\alpha \cup B_\beta$ of $2k$ inputs with probability at most $2^{-2k} + 2^{-(m-n)}$. The number of functions ϕ that assume the same values on the corresponding pairs of elements $a \in A_\alpha$ and $b \in A_\beta$ is 2^k. Thus, the probability that f is not r-mixed is at most $\binom{n}{r} 2^{2r} (2^{-k} + 2^{-(m-n-k)})$. If $m = (7+\delta)n$ for any $\delta > 0$, then this probability tends to 0 as n grows. □

By definition, each function $F_{x,y} : \{0,1\}^n \to \{0,1\}$, for $x, y \in \{0,1\}^m$, can be computed by a Boolean circuit of size $\mathrm{poly}(n,m)$. Given the coefficients of the lexicographically first irreducible degree-m polynomial over GF(2), one can check in linear space whether $F_{x,y}$ is r-mixed, for any given $x, y \in \{0,1\}^m$. The idea is to use the "brute-force" algorithm which enumerates all possible subsets of size r of the set $\{x_1, \ldots, x_n\}$, and checks whether any two distinct assignments to the selected r variables result in the same subfunction in the remaining $n - r$ variables. Clearly, the coefficients of the lexicographically first irreducible polynomial of degree m over GF(2) can be found in $O(m)$ space (by a "brute-force" algorithm). It then follows from Theorem 8 that we can find an r-mixed function, for $r = n - \lceil \log n \rceil - 2$, in LINSPACE by picking the lexicographically first string xy of $2m$ bits that determines such a function. By Lemma 4, this function will have the optimal lower bound for 1-b.p.'s, $\Omega(2^n/n)$. Thus, we have the following corollary.

Corollary 9. *The class* LINSPACE *contains a family $f = \{f_n\}_{n \geq 0}$ of Boolean functions f_n that require 1-b.p.'s of size at least $\Omega(2^n/n)$.*

We should point out that any of the three constructions of small (ε, k)-independent spaces in [2] could be used in the same manner as described above to obtain an r-mixed Boolean function computable in LINSPACE $\cap$ P/poly, for $r = n - \lceil \log n \rceil - 2$. Applying Lemma 3, we can obtain an r-mixed function with the same value of r.

Theorem 10. *There are $m \in O(\log n)$ and $l \in \mathrm{poly}(n)$ for which the probability that a Boolean n-variable function f computed by the random formula $\mathbf{F}(n,m,l)$ defined in (1) is r-mixed, for $r = n - \lceil \log n \rceil - 2$, tends to 1 as n grows.*

Proof. Proceeding as in the proof of Theorem 8, with Lemma 3 applied instead of Lemma 2, we obtain that the probability that f is not r-mixed is at most $\binom{n}{r} 2^{2r} (2^{-k} + 2^{-(l2^{-m}-k)})$. If $m = \lceil \log n \rceil + 3$ and $l = (6+\delta)n^2$ for any $\delta > 0$, then this probability tends to 0 as n grows. □

Corollary 11. *There exists a Boolean n-variable function computable by a polynomial-size depth 3 formula over the basis* $\{\&, \oplus, 1\}$ *that requires a 1-b.p. of size at least* $\Omega(2^n/n)$ *for all sufficiently large n.*

4. Constructing (*s, n, q*)-complete functions

Let us take a look at the probabilistic proof (as presented in [17]) of the existence of a matrix A and a function ψ with the properties assumed in Lemma 6. Suppose that a $t \times n$ matrix A over GF(2) and a function $\psi : \{0,1\}^t \to \{1,2,\ldots,n\}$ are chosen uniformly at random. For a fixed $t \times s$ submatrix B of A, if $\operatorname{rank}(B) < r$, then there is a set of at most $r-1$ columns in B whose linear span contains each of the remaining $s-r+1$ columns of B. For a fixed set R of such $r-1$ columns in B, the probability that each of the $s-r+1$ vectors chosen uniformly at random will be in the linear span of R is at most $(2^{r-1}/2^t)^{s-r+1}$. Thus, the probability that the matrix A is "bad" is at most $\binom{n}{s}\binom{s}{r-1}2^{-(t-r+1)(s-r+1)}$.

For a fixed affine subspace H of $\{0,1\}^t$ of dimension r and a fixed $1 \leqslant i \leqslant n$, the probability that the range of ψ restricted to H does not contain i is at most $(1-1/n)^{2^r}$. The number of different affine subspaces of $\{0,1\}^t$ of dimension r is at most $2^{(r+1)t}$; the number of different i's is n. Hence the probability that ψ is "bad" is at most $2^{(r+1)t} n(1-1/n)^{2^r} \leqslant 2^{(r+1)t} n \mathrm{e}^{-2^r/n}$.

An easy calculation shows that setting $s = \lceil (2+\delta)\log n \rceil$, $t = \lceil (3+\delta)\log n \rceil$, and $r = \lceil \log n + 2\log\log n + b \rceil$, for any $\delta > 0$ and sufficiently large b (say, $b = 3$ and $\delta = 0.01$), makes both the probability that A is "bad" and the probability that ψ is "bad" tend to 0 as n grows.

Theorem 12. *There are* $d_1, d_2, d_3 \in \mathbb{N}$ *such that every* $(2^{-d_1 \log^3 n}, d_2 \log^3 n)$*-independent sample space over* n^{d_3}*-bit strings contains both matrix A and function* ψ *with the properties as in Lemma 6, for* $s, r, t \in \mathrm{O}(\log n)$.

Proof. We observe that both probabilistic arguments used only partial independence of random bits. For A, we need a tn-bit string coming from an (ε, k)-independent sample space with $k = ts$ and $\varepsilon = 2^{-c_1 \log^2 n}$, for a sufficiently large constant c_1.

Indeed, for a fixed $t \times s$ submatrix B of A and a fixed set R of $r-1$ columns in B, the number of "bad" $t \times s$-bit strings α filling B so that the column vectors in R contain in their linear span all the remaining $s-r+1$ column vectors of B is at most $2^{(r-1)t} 2^{(r-1)(s-r+1)} = 2^{(r-1)(s+t-r+1)}$. If A is chosen from the (ε, k)-independent sample space with ε and k as above, then the probability that some fixed "bad" string α is chosen is at most $2^{-ts} + \varepsilon$. Thus, in this case, the probability that A is "bad" is at most

$$\binom{n}{s}\binom{s}{r-1}\left(2^{-(t-r+1)(s-r+1)} + \varepsilon 2^{(r-1)(s+t-r+1)}\right).$$

Choosing the same s, t, and r as in the case of fully independent probability distribution, one can make this probability tend to 0 as n grows, by choosing sufficiently large c_1.

Similarly, for the function ψ, we need a $2^t\lceil\log n\rceil$-bit string from an (ε,k')-independent sample space with $k'=c_2\log^3 n$ and $\varepsilon=2^{-c_3\log^3 n}$, for sufficiently large constants c_2 and c_3. Here we view the truth table of ψ as a concatenation of 2^t binary strings of length $\lceil\log n\rceil$, where each $\lceil\log n\rceil$-bit string encodes a number from $\{1,\ldots,n\}$. The proof, however, is slightly more involved in this case, and depends on Lemma 7.

Let s, r, and t be the same as before. For a fixed affine subspace $H\subseteq\{0,1\}^t$ of dimension r, such that $H=\{a_1,\ldots,a_l\}$ for $l=2^r$, and for a fixed $1\leqslant i\leqslant n$, let $\mathscr{E}_j$, $1\leqslant j\leqslant l$, be the event that $\psi(a_j)=i$ when ψ is chosen from the (ε,k')-independent sample space defined above. Then Lemma 7 applies with $\theta=2^{-\lceil\log n\rceil}$ and $k=k'/\log n=c_2\log^2 n$, yielding that the probability that ψ misses the value i on the subspace H is

$$\Pr\left[\bigwedge_{j=1}^{l}\mathscr{E}_j\right]\leqslant \mathrm{e}^{-2^{r-\lceil\log n\rceil}}+\binom{2^r}{k+1}(\varepsilon k+2^{-k\lceil\log n\rceil}). \tag{6}$$

It is easy to see that the first term on the right-hand side of (6) is at most $\mathrm{e}^{-4\log^2 n}$ (when $b=3$ in r). We need to bound from above the remaining two terms: $\binom{2^r}{k+1}2^{-k\lceil\log n\rceil}$ and $\binom{2^r}{k+1}\varepsilon k$. Using Stirling's formula, one can show that the first of these two terms can be made at most $2^{-4\log^2 n}$.

Indeed, using the fact that $\binom{m}{l}\leqslant m^l/(l!)$, we have

$$\begin{aligned}\binom{2^r}{k+1}2^{-k\lceil\log n\rceil}&\leqslant\frac{2^{rk+r-k\lceil\log n\rceil}}{(k+1)!}\\&\leqslant\frac{2^{2k\log\log n+(bk+\lceil\log n\rceil+2\log\log n+b)}}{k!}\\&\leqslant\frac{(\log^2 n)^k 2^{b'k}}{k!},\end{aligned}$$

for some constant $b'>b$. By Stirling's formula, $k!\geqslant(k/e)^k$. Using this lower bound on $k!$, we can continue our sequence of inequalities as follows:

$$\begin{aligned}\frac{(\log^2 n)^k 2^{b'k}}{k!}&\leqslant\frac{(\log^2 n)^k 2^{b'k}}{(c_2\log^2 n/e)^k}\\&=\frac{2^{b'k}}{(c_2/e)^k}\\&=2^{(b'-\log c_2/e)k}.\end{aligned}$$

By choosing c_2 sufficiently larger than b', we obtain the required bound on $\binom{2^r}{k+1}2^{-k\lceil\log n\rceil}$.

Having fixed c_2, we can also ensure that our second term, $\binom{2^r}{k+1}\varepsilon k$, is at most $2^{-4\log^2 n}$. Indeed, we have

$$\begin{aligned}\binom{2^r}{k+1}\varepsilon k &\leqslant \frac{2^{r(k+1)}\varepsilon k}{(k+1)!}\\ &\leqslant 2^{2c_2\log^3 n - c_3\log^3 n}.\end{aligned}$$

By choosing c_3 sufficiently bigger than c_2, we obtain the required bound.

It is then straightforward to verify that the probability that ψ misses at least one value i, $1\leqslant i\leqslant n$, on at least one affine subspace of dimension r tends to 0 as n grows. □

Using the sample space Pow_N^{2m} with $N = tn \in \mathrm{O}(n\log n)$ and $m \in \mathrm{O}(\log^2 n)$, we can find a matrix A with the required properties in $\mathrm{DTIME}(2^{\mathrm{O}(\log^2 n)})$ as follows: we find the coefficients of the lexicographically first irreducible degree-m polynomial over GF(2) (by a "brute-force" algorithm in time $2^{\mathrm{O}(m)}$), and then we search through all elements of the sample space and check whether any of them yields a desired matrix. Analogously, we can find the required function ψ in $\mathrm{DTIME}(2^{\mathrm{O}(\log^3 n)})$, by considering, e.g., $\mathrm{Pow}_{N'}^{2m'}$ with $N' = 2^t\lceil\log n\rceil$ and $m' \in \mathrm{O}(\log^3 n)$. Thus, constructing both A and ψ can be carried out in quasipolynomial time.

Given the corresponding advice strings of $\mathrm{O}(\log^3 n)$ bits (which include the coefficients of the irreducible polynomials of degrees m and m' as well as two binary strings of lengths $2m$ and $2m'$), ψ is computable in time polylog(n) and all elements of A can be computed in time $n\,\mathrm{polylog}(n)$. So, in this case, the function $\phi(x) = \psi(Ax)$ is computable in quasilinear time. Hence, by "hard-wiring" good advice strings, we get the function $f_n(x) = x_{\phi(x)}$ computable by quasilinear-size circuits, while, by Lemmas 5 and 6, f_n requires 1-b.p.'s of size at least $2^{n-(5+\varepsilon)\log n}$, for any $\varepsilon > 0$ and sufficiently large n; these parameters appear to be better than those in [3]. By making the advice strings a part of the input, we obtain a function in QL that requires 1-b.p.'s of size at least $2^{n-\mathrm{O}(\log^3 n)}$.

These results are summarized in the following corollary.

Corollary 13.

- *The class* QP *contains a family* $f = \{f_n\}_{n\geqslant 0}$ *of Boolean functions that require* 1-*b.p.'s of size at least* $2^{n-\mathrm{O}(\log n)}$.
- *The class* QL *contains a family* $f = \{f_n\}_{n\geqslant 0}$ *of Boolean functions that require* 1-*b.p.'s of size at least* $2^{n-\mathrm{O}(\log^3 n)}$.

5. Constructing $(n - \mathrm{O}(\log n))$-mixed Boolean functions

In this section, we point out that the method used above to construct an (s,n,q)-complete Boolean function could be also used to construct an r-mixed Boolean function

for $r = n - O(\log n)$. This can be achieved by derandomizing Savický's [16] modification of the procedure in [17]. For completeness, we present this modification below.

Theorem 14 (Savický [16]). *Let A be a $t \times n$ matrix over* GF(2) *with every $t \times s$ submatrix of rank at least r. Let $\psi : \{0,1\}^t \to \{1,\ldots,n\}$ be a mapping such that, for every affine subset $H \subseteq \{0,1\}^t$ of dimension at least $r-1$ and every vector $\Delta \in \{0,1\}^t$ such that $H \cap (H + \Delta) = \emptyset$, the following holds: for each $1 \leqslant i \leqslant n$, there is a $w \in H$ such that $\psi(w) = \psi(w + \Delta) = i$. Then the function $f(x) = x_{\phi(x)}$ is $(n-s)$-mixed, where $\phi(x) = \psi(Ax)$.*

Proof. Fix an arbitrary subset $I \subseteq \{1,\ldots,n\}$ of size $n - s$. Let $a, b \in \{0,1\}^{n-s}$ be two distinct assignments to the variables x_i for $i \in I$. Let y be the vector of x_j's for $j \in \bar{I}$, i.e., the vector of the unset variables x_j. Let A_1 be the $t \times (n-s)$-submatrix of A that contains the columns of A with indices in I; let A_2 be the $t \times s$-submatrix containing the remaining columns of A. Thus, we have $\phi(a, y) = \psi(A_1 a + A_2 y)$ and $\phi(b, y) = \psi(A_1 b + A_2 y)$.

Let $\Delta = A_1(a + b)$ and let $U \subseteq \{0,1\}^t$ be the linear space of all possible products $A_2 y$. Let i be the index such that $a_i \neq b_i$. To prove that $f(x)$ is $(n-s)$-mixed, it suffices to show that there is a $u \in U$ such that $\psi(A_1 a + u) = \psi(A_1 b + u) = i$.

By the first assumption of the theorem, the dimension of U is at least r. It is easy to see that there exists a linear subspace $U' \subset U$ of dimension at least $r-1$ such that $\Delta \notin U'$. Define the affine space $H = A_1 a + U'$. Obviously, we have $H + \Delta = A_1 b + U'$. Since $\Delta \notin U'$, we obtain that $H \cap (H + \Delta) = \emptyset$. Now, applying the second assumption of the theorem, we conclude that the required vector $u \in U$ exists. □

Theorem 15 (Savický [16]). *The matrix A and the function ψ as required by Theorem* 14 *exist for $t, s, r \in \Theta(\log n)$ and for all sufficiently large n.*

Proof. As in our argument at the beginning of Section 4, we obtain that the probability that a random $t \times n$ matrix A is "bad" is at most $\binom{n}{s}\binom{s}{r-1} 2^{-(t-r+1)(s-r+1)}$.

For a fixed affine subspace H of $\{0,1\}^t$ of dimension $r-1$, a fixed vector $\Delta \in \{0,1\}^t$ such that $H \cap (H + \Delta) = \emptyset$, and a fixed $1 \leqslant i \leqslant n$, the probability that a random function ψ *fails* to satisfy the condition $\psi(w) = \psi(w + \Delta) = i$ for *every* $w \in H$ is at most $(1 - 1/n^2)^{2^{r-1}}$. The number of different affine subspaces of $\{0,1\}^t$ of dimension $r-1$ is at most 2^{rt}; the number of different Δ's is at most 2^t; and the number of different i's is n. Hence the probability that ψ is "bad" is at most $2^{(r+1)t} n (1 - 1/n^2)^{2^{r-1}} \leqslant 2^{(r+1)t} n e^{-2^{r-1}/n^2}$.

Let us set $s = \lceil (3+\delta) \log n \rceil$, $t = \lceil (5+\delta) \log n \rceil$, and $r = \lceil 2 \log n + 2 \log \log n + b \rceil$, for any $\delta > 0$ and sufficiently large b (say, $b = 5$ and $\delta = 0.01$). Then it is easy to verify that both the probability that A is "bad" and the probability that ψ is "bad" tend to 0 as n grows. □

This probabilistic argument can also be derandomized.

Theorem 16. *There are $d_1, d_2, d_3 \in \mathbb{N}$ such that every $(2^{-d_1 \log^3 n}, d_2 \log^3 n)$-independent sample space over n^{d_3}-bit strings contains both matrix A and function ψ with the properties as in Theorem* 14, *for $s, r, t \in O(\log n)$.*

Proof. The proof is very similar to that of Theorem 12. We use the values for s, r, t as given in the proof of Theorem 15. For the matrix A, the argument is exactly the same as in the proof of Theorem 12.

For the function ψ, the difference is in the definition of the family of events $\mathscr{E}_j$. Namely, for an affine subspace $H \subseteq \{0,1\}^t$ of dimension $r-1$, such that $H = \{a_1, \ldots, a_l\}$ for $l = 2^{r-1}$, for a fixed $\Delta \in \{0,1\}^t$ such that $H \cap (H + \Delta) = \emptyset$, and for a fixed $1 \leqslant i \leqslant n$, we define $\mathscr{E}_j$, $1 \leqslant j \leqslant l$, to be the event that $\psi(a_j) = \psi(a_j + \Delta) = i$ when ψ is chosen from the (ε, k')-independent sample. Then Lemma 7 applies with $\theta = 2^{-2\lceil \log n \rceil}$ and $k = k'/(2 \log n)$. The rest of the argument is essentially the same. □

As in the previous section we can construct both A and ψ in quasipolynomial time, by using an efficient construction of almost k-wise independent sample spaces. Hence, we obtain the following.

Corollary 17. *The class* QP *contains a family* $f = \{f_n\}_{n \geqslant 0}$ *of* $(n - \mathrm{O}(\log n))$*-mixed Boolean functions* f_n.

6. Concluding remarks

We have shown how the well-known constructions of small ε-biased sample spaces [12,10,2] can be directly used to obtain Boolean functions that are exponentially hard for 1-b.p.'s. One might argue, however, that the hard Boolean functions constructed in Section 3 and 4 are not "explicit" enough, since they are defined as the lexicographically first functions in certain search spaces. It would be interesting to find a Boolean function in P or NP with the optimal lower bound $\Omega(2^n/n)$ for 1-b.p.'s. The problem of constructing a polynomial-time computable r-mixed Boolean function with r as large as possible is of independent interest; at present, the best such function is given in [17] for $r = n - \Omega(\sqrt{n})$. A related open question is to determine whether the minimum number of bits needed to specify a Boolean function with the optimal lower bound for 1-b.p.'s, or an r-mixed Boolean function for $r = n - \lceil \log n \rceil - 2$, can be sublinear.

Acknowledgements

I am indebted to Alexander Razborov for bringing [12] to my attention. I would like to thank Stephen Cook and Petr Savický for their comments on a preliminary version of this paper, and Dieter van Melkebeek for helpful discussions. I am particularly thankful to Petr Savický for the permission to include his construction of $(n - \mathrm{O}(\log n))$-mixed Boolean functions (Theorems 14 and 15). I am very grateful to the anonymous referee for insightful comments and helpful suggestions. Finally, I want to express my sincere gratitude to Stephen Cook for his constant encouragement and support.

References

[1] M. Ajtai, L. Babai, P. Hajnal, J. Komlós, P. Pudlak, V. Rödl, E. Szemerédi, G. Turán, Two lower bounds for branching programs, in: Proc. Eighteenth Annu. ACM Symp. on Theory of Computing, 1986, pp. 30–38.

[2] N. Alon, O. Goldreich, J. Håstad, R. Peralta, Simple constructions of almost k-wise independent random variables, Random Struct. Algorithms 3 (3) (1992) 289–304 (preliminary version in FOCS'90).

[3] A.E. Andreev, J.L. Baskakov, A.E.F. Clementi, J.D.P. Rolim, Small pseudo-random sets yield hard functions: new tight explicit lower bounds for branching programs. Electron. Colloq. Computational Complexity TR97-053, 1997.

[4] B. Bollig, I. Wegener, A very simple function that requires exponential size read-once branching programs, Inform. Process. Lett. 66 (1998) 53–57.

[5] P.E. Dunne, Lower bounds on the complexity of one-time-only branching programs, in: L. Budach (Ed.), Proc. 2nd Int. Conf. on Fundamentals of Computation Theory, Lecture Notes in Computer Science, vol. 199, Springer, Berlin, 1985, pp. 90–99.

[6] A. Gal, A simple function that requires exponential size read-once branching programs, Inform. Process. Lett. 62 (1997) 13–16.

[7] R. Impagliazzo, A. Wigderson, P = BPP if E requires exponential circuits: derandomizing the XOR Lemma, in: Proc. Twenty-Ninth Annu. ACM Symp. on Theory of Computing, 1997, pp. 220–229.

[8] S. Jukna, Entropy of contact circuits and lower bound on their complexity, Theoret. Comput. Sci. 57 (1988) 113–129.

[9] M. Krause, C. Meinel, S. Waack, Separating the eraser Turing machine classes L_e, NL_e, $co-NL_e$ and P_e, Theoret. Comput. Sci. 86 (1991) 267–275.

[10] J. Naor, M. Naor, Small-bias probability spaces: efficient constructions and applications, SIAM J. Comput. 22 (4) (1993) 838–856 (preliminary version in STOC'90).

[11] S. Ponzio, A lower bound for integer multiplication with read-once branching programs, SIAM J. Comput. 28 (3) (1999) 798–815 (preliminary version in STOC'95).

[12] A.A. Razborov, Bounded-depth formulae over $\{\&, \oplus\}$ and some combinatorial problems, in: S.I. Adyan (Ed.), Problems of Cybernetics, Complexity Theory and Applied Mathematical Logic, VINITI, Moscow, 1988, pp. 149–166 (in Russian).

[13] A.A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, in: L. Budach (Ed.), Proc. 8th Int. Conf. on Fundamentals of Computation Theory, Lecture Notes in Computer Science, vol. 529, Springer, Berlin, 1991, pp. 47–60.

[14] A.A. Razborov, S. Rudich, Natural proofs, J. Comput. System Sci. 55 (1997) 24–35.

[15] P. Savický, Improved Boolean formulas for the Ramsey graphs, Random Struct. Algorithms 6 (4) (1995) 407–415.

[16] P. Savický, personal communication, January 1999.

[17] C.E. Shannon, The synthesis of two-terminal switching circuits, Bell Systems Tech. J. 28 (1) (1949) 59–98.

[18] J. Simon, M. Szegedy, A new lower bound theorem for read-only-once branching programs and its applications, in: J.-Y. Cai (Ed.), Advances in Computational Complexity, AMS-DIMACS Series, 1993, pp. 183–193.

[19] P. Savický, S. Zák, A large lower bound for 1-branching programs, Electron. Colloq. on Computational Complexity TR96-036, 1996.

[20] I. Wegener, On the complexity of branching programs and decision trees for clique function, J. Assoc. Comput. Mach. 35 (1988) 461–471.

[21] S. Zak, An exponential lower bound for one-time-only branching programs, in: Proc. Eleventh Int. Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, vol. 176, Springer, Berlin, 1984, pp. 562–566.